A Dual Encryption Protocol for Scalable Secure Multicasting

Lakshminath R. Dondeti
University of Nebraska-Lincoln

Sarit Mukherjee
Panasonic Information & Networking Technology Lab, sarit@research.panasonic.com

Ashok K. Samal
University of Nebraska-Lincoln, asamal1@unl.edu

A Dual Encryption Protocol for Scalable Secure Multicasting

Lakshminath R. Dondeti
University of Nebraska-Lincoln
115 Ferguson Hall, Lincoln, NE 68588-0115
ldondeti@cse.unl.edu

Sarit Mukherjee
Panasonic Information &
Networking Technology Lab
2 Research Way, Princeton, NJ 08540
sarit@research.panasonic.com

Ashok Samal
University of Nebraska-Lincoln
115 Ferguson Hall, Lincoln, NE 68588-0115
samal@cse.unl.edu

Abstract

In this paper we propose a dual encryption protocol for scalable secure multicasting. Multicasting is a scalable solution for group communication. It however, poses several unique security problems. We use hierarchical subgrouping to achieve scalability. Third party hosts or members of the multicast group are designated as subgroup managers. They are responsible for secret key distribution and group membership management at the subgroup level. Unlike existing secure multicast protocols, our protocol need not trust the subgroup managers with the distribution of data encryption keys. The dual encryption protocol proposed in this paper distributes encrypted data encryption keys via subgroup managers. We also present a classification of the existing secure multicast protocols, compare their relative merits and show the advantages of our protocol.

1. Introduction

With the widespread use of the Internet, securing data transmissions is an important requirement for many applications. Several protocols exist to address security in data networks with respect to unicasting [6, 10]. Unfortunately, these protocols cannot be easily extended to protect multicast data. Multicasting poses several problems that do not come up in securing unicast data transfers [2]. First, multicast addresses are not private which enables any interested host to join the multicast session without any hindrance. Next, multicast data is transmitted over many channels of the network which presents multiple opportunities for attacks such as eavesdropping. Furthermore, any host in the Internet can send irrelevant data to the multicast group, which may cause congestion. The universal knowledge of multicast addresses also allows any host to pose as a member of the group, thereby allowing it to gain access to the multicast data. Finally, adversaries can possibly disrupt the multicast protocol itself by posing as legitimate members of the group.

Multicasting is a scalable way of transmitting data to a group hosts and any secure multicasting protocol must be scalable as well. A secure group communication protocol must provide group membership control, secure key distribution, and secure data transfer. If the multicast group membership is dynamic, i.e., if the group members join and leave during the course of a multicast session, the secret keys need to be updated accordingly. In other words, members of a multicast session must not be able to access the multicast data transmitted before their membership has begun or after their membership has expired. Scalability in this context implies that the overhead involved in key updates, data transmission and encryption must be independent of the size of the multicast group. The other requirement of scalability is that the addition or removal of a host from the group must not affect all the members of the group. The second rule is called "1 affects n" scalability problem [11].

Several protocols have been proposed to support secure multicasting [1, 4, 7, 9, 11, 14, 15]. Based on the corresponding key distribution protocols, we can broadly classify them into three categories, viz., centralized flat schemes, distributed flat schemes and hierarchical schemes [4]. Centralized flat schemes [3, 4, 8, 9] consist of a single entity distributing the encryption keys to the group members. These schemes suffer from the 1 affects n scalability problem. The distributed flat schemes trust all the group members equally [4]. Members joining early create and distribute the encryption keys. Trusting all the members makes this protocol vulnerable to security attacks from inside the group. Finally, the hierarchical schemes [1, 4, 11, 14, 15] distribute encryption keys via a distribution tree. The schemes proposed by Caronni *et. al* [4], Wallner *et. al* [14] and Wong *et. al* [15] suffer from the 1 affects n scalability problem. The other hierarchical schemes entrust the internal nodes of the tree with the distribution of the encryption keys. But they offer no mechanism to hide multicast data from these third party entities.

We propose a dual encryption protocol (DEP) for scalable secure multicasting which supports one-to-many group communication. We use hierarchical subgrouping of multicast members to address scalability. Each subgroup is managed by a subgroup manager (SGM) which assists in key distribution as well as group access control. We distinguish between *participants* and *members* of the multicast group. Members of the multicast group are leaf nodes and internal nodes (SGMs) in the key distribution tree, that are entitled to the multicast data. On the other hand, participants of the multicast group are SGMs that assist in enforcing the secure multicast protocol without having any access to the multicast data. The dual encryption scheme enables our protocol to hide multicast data from the participants.

The rest of this paper is organized as follows. We describe the dual encryption protocol for scalable secure multicasting in Section 2. Section 3 compares existing secure multicast protocols to our protocol. The final section lists our conclusions and describes future directions in this area.

2. Anatomy of the dual encryption protocol

In this section we describe the dual encryption protocol for scalable secure multicasting. Our protocol supports secure one-to-many group communication, dynamic group membership and is scalable. We use hierarchical subgrouping of multicast members to address scalability. Each subgroup is managed by a subgroup manager (SGM). SGMs are either routers or hosts in the network that can handle the workload of managing a subgroup of the multicast group. We assume that the SGMs conform to the secure multicast protocol and do not actively participate in disrupting it. We distinguish between *participants* and *members* of the group. Members of the group are end-hosts or SGMs that are entitled to the multicast data. On the other hand, participants of the group are SGMs that assist in enforcing the secure multicast protocol without having any access to the multicast data. With this distinction, it is possible to have SGMs assist in the secure multicast protocol without getting access to multicast data.

We use two sets of encryption keys that assist in secure distribution of data encryption keys to multicast members. The first set of keys called local subgroup keys (LS) are used by SGMs to distribute encrypted data encryption keys to their corresponding subgroup members. The second set of keys called top level key encrypting keys (KEK) are used by the sender to hide data encryption keys from participant SGMs. We classify the members and participants of the multicast group into *key groups*. The members in each key group get access to the same KEK. Nodes of each subtree rooted at one of the sender's children belong to the same key group. Nodes of different subtrees rooted at the sender's children may belong to the same key group. The number of key groups however is limited by the number of SGMs among the sender's children. Our protocol uses public-key [10, 12, 13] encryption for securely distributing the top level KEKs and the subgroup keys.

Authentication Information	**Authorization Information**
Host Name	Multicast Group Name
Host Identifier	Multicast Group Identifier
Host Public Key	Membership Duration [Start time, Finish Time]
Signed by Certification Authority	

Figure 1. Capability certificate

Table 1. Notation used in this paper

s	Sender
SGM	Subgroup manager
LS_i	Local subgroup key managed by SGM i
$\mathcal{M}_i$	Set of subgroup members of SGM i
$\mathcal{K}_i$	Set of nodes of the key distribution tree in the key group i
KEK_i	Key encrypting key corresponding to the key group $\mathcal{K}_i$
DEK	Data encryption key used in encrypting multicast data
CC_x	Capability Certificate of x
AC_x	Authorization Certificate issued to x by s
KU_x	Public-key of x
KR_x	Private-key of x
EP	Public-key encryption
ES	Secret-key encryption
HV	Hash value
$x \rightarrow y$: w	x sends "w" to y

We use capability certificates (see Figure 1) to enforce group access control. For large multicast groups, access control lists can be very large. Furthermore, we may not know all the group members in advance. Our protocol requires that all the members obtain a capability certificate from designated certification authorities. These certificates authenticate hosts and authorize them to be members of the multicast group. The authorization information also includes the time duration for which the group member is entitled to multicast data. The sender and the SGMs verify the capability certificates before distributing encryption keys to group members. We list the notation used in the remainder of the protocol description, in Table 1.

2.1. Initial key distribution

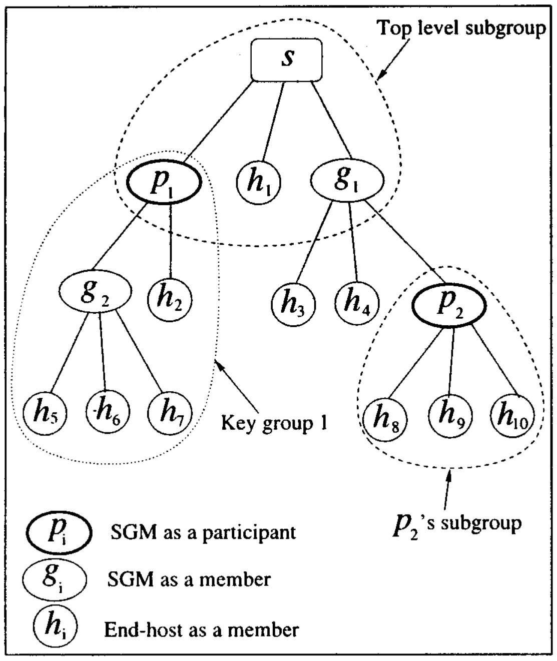

Figure 2. Key distribution tree

Figure 2 illustrates the idea of a secure multicast key distribution tree. As depicted in the figure, members of the multicast group are nodes of a key distribution tree. The key distribution tree can be either an extension of a multicast data distribution tree (e.g. DVMRP [5]) or a virtual tree at the application level. Subgroup managers are represented by the non-leaf nodes and the children of each of these non-leaf nodes are subgroup members of the corresponding non-leaf node. Each SGM is responsible for generating a secret key and sharing it with all the corresponding subgroup members in a secure fashion. For instance in Figure 2, p_1 shares the subgroup key LS_{p_1} with its children, g_2 and h_2. The sender generates the top level KEKs and a local subgroup key for the top level subgroup. Recall that the KEKs are used to hide data encryption keys from the participants of the multicast group. One KEK is generated corresponding to each of the key groups. These keys are distributed to the multicast members by the sender. A KEK is shared by all the nodes in a key group, that are members of the multicast group. To illustrate the concept of key groups, we use the key distribution tree shown in Figure 2. There could be at most two key groups, corresponding to each of the sender's children that are SGMs, viz., p_1 and g_1. h_1 could belong to either one of the key groups.

All the members and participants of the multicast group must be aware of the key group they belong to. We delegate the responsibility of propagating this information to the subgroup managers. The sender assigns and distributes key group ids to the SGMs that are members of the top level subgroup. Each SGM disseminates its key group id to its subgroup members when they join the group. Thus, all the members and participants of the multicast session are aware of the corresponding key group id. Section 2.2 illustrates the distribution of the secret keys as a part of the join protocol of the dual encryption scheme.

2.2. Join protocol

When a new host h wants to join the secure multicast group, it sends a message which includes its capability certificate, to all the SGMs of the group and waits for the first positive response. SGMs that can handle the additional workload of another member in their subgroup respond to this request. They first verify the capability certificate to decide whether to approve or deny the request. Assuming that the request is approved, each SGM validating the request sends a message comprising of its identity and its key group identity. h chooses the first positive response it receives to pick the subgroup it is going to join. Let us say p_1 (refer to Figure 2) responds first. It sends its identity, p_1, and its key group identity, say $\mathcal{K}_1$ to h. Since p_1 replied first, h chooses p_1 as its subgroup manager.

The enrolling host then sends a packet with its capability certificate, the responding SGM's identity and the corresponding key group identity to the sender of the multicast data. The sender uses the capability certificate to decide whether h is an authorized member of the multicast group. It also checks to see if h has previously requested to join the multicast. This last verification is to guard against a misbehaving host trying to join multiple subgroups simultaneously. After the new host's membership is validated, the sender generates an authorization certificate (see Figure 3) for h. The authorization certificate contains the new host's identity (h), the corresponding SGM's identity and the key group identity. The sender signs the certificate with its private key. The authorization certificate is an authentic record of the new host's affiliation to the multicast group. Next, the sender sends the authorization certificate and the top level KEK to the joining host. The KEK sent to h corresponds to its key group identity. The sender encrypts the authorization certificate and the KEK with h's public key for secrecy. For example in Figure 2 the sender sends the packet,

$EP_{KUh}[\ EP_{KRs}[AC_h],\ EP_{KRs}[KEK_1]\ ]$. Note that the sender signs the authorization certificate and the KEK separately. This allows h to produce the signed authorization certificate without having to disclose the KEK. Finally, the sender updates its multicast membership database with the new host's authorization certificate. The membership database is used when the sender refreshes the KEKs.

Host Id	Multicast Group Id
Host Public Key	Key Group Id
Membership Duration	SGM Id
Signed by the Sender	

Figure 3. Authorization certificate

In the final phase of the join protocol, the host sends its signed authorization certificate to its SGM. The subgroup manager first adds the new host to its subgroup members' list. The SGM then changes its subgroup key, signs it, encrypts it with h's public key and sends it to h. The SGM's signature guards against masquerading attacks. The subgroup key is changed to keep the new host from decrypting multicast data sent before it joined the group. Separately, the SGM multicasts its signed new subgroup key to all its subgroup members, encrypted with the old subgroup key. In our example, p_1 sends $EP_{KUh}[EP_{KRp_1}[LS'_{p_1}]]$ to h and $ES_{LS_{p_1}}[EP_{KRp_1}[LS'_{p_1}]]$ to g_2 and h_2. Table 2 lists the steps in the join protocol. In the table, h joins the group at an SGM g, which belongs to the key group $\mathcal{K}_i$.

Table 2. Steps in the join protocol

(1)	$h \rightarrow$ SGMs:	CC_h
(2)	$g \rightarrow h$:	SGM Id(g), Key group Id($\mathcal{K}_i$)
(3)	$h \rightarrow s$:	CC_h, g, $\mathcal{K}_i$
(4)	$s \rightarrow h$:	$EP_{KUh}[\ EP_{KRs}[AC_h], EP_{KRs}[KEK_1]\ ]$
(5)	$h \rightarrow g$:	$EP_{KRs}[AC_h]$
(6)	$g \rightarrow h$:	$EP_{KUh}[\ EP_{KRg}[LS'_g]\ ]$
(7)	$g \rightarrow \mathcal{M}_g$:	$ES_{LS_g}[\ EP_{KRg}[LS'_g]\ ]$

Finally, we want to stress the importance of the use of authorization certificates. They eliminate the possibility of an adversary with a valid capability certificate gaining access to all the keys managed by the sender and all the SGMs in the multicast group. In our protocol, the sender checks for duplicate joins by the same host before issuing an authorization certificate. These certificates authorize the joining host to gain access to only one local subgroup key.

We now describe the join protocol followed by subgroup managers. SGMs that are members of the multicast group follow the join protocol described earlier. The only change is that the sender updates its SGM database. Recruiting a participant SGM, however, is more complex. The sender first verifies if the participant SGM is an ex-member of the multicast group. If the participant SGM is in the membership database, the corresponding KEK needs to be updated. To change a KEK, the sender sends a message to all the members which hold that KEK, asking them to request the new KEK. The members which need the new KEK respond with their authorization certificates. The sender verifies the authorization certificates and constructs a list of members authorized to receive the updated KEK. The sender then changes the KEK, signs it, and encrypts it with the public keys of all the members in the list. It then multicasts all the encrypted KEKs to the multicast group. Each member waiting for the new KEK decrypts the encrypted KEK intended for it. Finally, the sender updates its membership database conforming to the authorization list it compiled above. After the verification process and possible modification of a KEK, the join process of a participant SGM follows the same protocol described earlier. The only exception is that a participant SGM does not receive a KEK.

Clearly, the process of changing a KEK is costly operation. However, KEKs need to be changed only when an ex-member of the multicast group wants to rejoin as a participant SGM. To avoid changing KEKs frequently, an application may deny the join request of a participant SGM if it is still in the membership database.

2.3. Secure communication

The sender generates a data encryption key (DEK) to be used in a conventional encryption algorithm [10, 13]. It sends the multicast data encrypted with the DEK to the group. Next, it computes a one-way hash function of the data and sends the hash value (HV) along with the DEK to multicast members securely. The members also compute the hash value of multicast data and compare it to the HV received, to verify the integrity of the data.

While the encrypted multicast data is sent through traditional multicast channels, the DEKs are distributed via the key distribution tree. We use the key distribution tree in Figure 2 to illustrate the DEK distribution. The sender generates a key distribution packet ($ES_{LS_s}[ES_{KEK_1}$[DEK, HV]], $ES_{LS_s}[ES_{KEK_2}$[DEK, HV]]), where LS_s is the subgroup key of the top level subgroup. Each of the sender's children decrypts its part of the key distribution packet. Each of them then encrypts its piece of the packet with the subgroup key they manage and multicasts the encrypted DEK to its children. In our example in Figure 2, p_1 multicasts the encrypted packet that contains $ES_{LS_{p_1}}[ES_{KEK_1}$[DEK, HV]], to g_2 and h_2. Similarly, other SGMs forward the encrypted DEK to their respective subgroup members. All the mem-

Table 3. Steps in the DEK distribution protocol

(1)	$s \rightarrow \mathcal{M}_s$:	$\text{ES}_{LS_s}[\text{ES}_{KEK_1}[\text{DEK, HV}]], \ldots, \text{ES}_{LS_s}[\text{ES}_{KEK_c}[\text{DEK, HV}]]$
(2)	$g_i \rightarrow \mathcal{M}_{g_i}$:	$\text{ES}_{LS_{g_i}}[\text{ES}_{KEK_i}[\text{DEK, HV}]]$

bers of the multicast group with a local subgroup key and the corresponding KEK acquire the DEK and HV. The DEK is used by the members to decrypt the multicast data and HV is used to verify the integrity of multicast data. Note that the SGMs that are also members of the multicast group will have access to the corresponding KEK. Other SGMs will just participate in the secure multicast protocol by managing their corresponding subgroup key and forwarding the encrypted DEK. Table 3 lists the steps in the DEK distribution protocol. In the table, we assume that there are c key groups and that SGM g_i, which is one of the sender's children, belongs to the key group $\mathcal{K}_i$.

2.4. Leave protocol

The membership of a multicast group member may expire as per the membership duration information in the capability certificate. It is also possible that either the sender or the corresponding SGM may have to expel a misbehaving member. In either case, the ex-member of the multicast session must not be able to decrypt the multicast data. To do that, the corresponding SGM changes the local subgroup key. It then encrypts the new subgroup key with the public keys of each of its children and multicasts that information to them. Each of the children decrypts its part of that message and extracts the updated subgroup key. Revisiting our example in Figure 2, if the host h_9 leaves the multicast group, the corresponding subgroup manager, p_2 changes the subgroup key and securely sends the new key to the hosts h_8 and h_{10} separately.

Note that the KEK known to the leaving host need not be changed right away. The sender can periodically change those keys depending on the frequency of hosts rejoining the group. Since any member needs to know both the corresponding subgroup key and the key encrypting key to decrypt the DEK, changing even one of them is sufficient. We list the the steps of the leave protocol in Table 4. In the table, we assume that h_i left from SGM g, where $\mathcal{M}_g = \{h_1, h_2, \ldots, h_m\}$ and that LS'_g is the new subgroup key.

Dual encryption of the DEK simplifies the removal of an SGM from the multicast group. All we need to do is to remove the SGM, find a replacement and notify the subgroup members of the change. Note that each SGM is a member of a subgroup managed by its parent. The parent SGM removes the leaving SGM, following a procedure identical to that of removing a member of the multicast group. The sender needs to locate another SGM that replaces the leaving SGM. After finding a replacement, the sender notifies the members of the subgroup managed by the leaving SGM about their new subgroup manager. The sender also updates its list of SGMs. The new SGM follows the join protocol to become either a participant or a member of the multicast group. After that, it generates the subgroup key and securely distributes that key to its subgroup members.

2.5. Key refresh

The sender and the SGMs refresh their keys periodically to guard against eavesdropping. To change the subgroup key, a subgroup manager follows the same procedure described in Section 2.4. In brief, the SGM changes the key, signs it and encrypts it with the public keys of all the subgroup members. It then locally multicasts the updated subgroup key to its subgroup members. Refreshing KEKs is a complex procedure and is expected to be done infrequently. The sender can change a KEK following the mechanism described in Section 2.2. In general, KEKs may be refreshed depending on the frequency of hosts rejoining the multicast group.

2.6. Tuning the number of key encrypting keys

The number of KEKs can be between zero and the number of SGMs in the top level subgroup. When the number of KEKs is *zero* all the SGMs automatically receive access to multicast data. The use of a single KEK gives us the capability of denying access of multicast data to SGMs. However, the KEK may need to be refreshed/updated more often since it is shared by all of the members. As the number of KEKs increase the refresh/update frequency decreases. The upper bound to the number of KEKs is the number of SGMs that are also members of the top level subgroup.

3. Previous research in scalable key distribution

We summarize the previous work done in the area of secure scalable key distribution in this section. Most of the previous work in the area of secure multicasting has been in key distribution. We can loosely classify the existing schemes as centralized flat key management, hierarchical key management, and distributed flat key management schemes [4].

Table 4. Steps in the leave protocol

(1)	$g \rightarrow \mathcal{M}_g$:	$\mathrm{EP}_{KUh_1}[\mathrm{EP}_{KRg}[LS'_g]], \ldots, \mathrm{EP}_{KUh_{i-1}}[\mathrm{EP}_{KRg}[LS'_g]], \mathrm{EP}_{KUh_{i+1}}[\mathrm{EP}_{KRg}[LS'_g]], \ldots, \mathrm{EP}_{KUh_m}[\mathrm{EP}_{KRg}[LS'_g]]$

Centralized flat key management. In this approach a single entity distributes the session key to all the group members. The protocol suggested in Elements of Trusted Multicasting [8] (ETM), distributes the session key encrypted with each of the group members' public keys after multicast data has been sent. In Group Key Management Protocol (GKMP) [9], the group manager shares a session traffic encrypting key (TEK) and key encrypting key (KEK) with each member. These keys are used in distributing the group TEK and the group KEK. The Perfectly-Secure Dynamic-Conference Key Distribution (PDKD) [3] protocol describes a secret-sharing scheme for secure group communication. Each host in any group of t members can compute a common key, on input its share of the secret and the identities of the other $t - 1$ hosts. In the Centralized Flat Key Management for Dynamic Multicast Groups (CFKM-DMG) [4] the group manager generates a TEK and $2W$ KEKs, where W represents the number of bits in any member's ID. For each bit in the ID, there are two KEKs. Each member receives the TEK and W KEKs corresponding to the values of each bit in its ID.

Hierarchical key management. In this approach, the members of the multicast group are part of a tree-like hierarchical structure. The intermediate nodes in the tree assist in key distribution. We can further classify these schemes into two different groups, viz., hierarchical node based approaches and hierarchical key based approaches. In the following, the first two schemes are hierarchical node based and the others are hierarchical key based schemes.

Scalable Multicast Key Distribution (SMKD) protocol [1] uses the Core Based Tree (CBT) architecture for multicasting, for key distribution purposes. The primary core creates a session key and a key encrypting key (KEK). The session key, and the KEK are distributed to the secondary core and subsequently to other nodes as they become part of the distribution tree.

Iolus [11] proposes the idea of hierarchical subgrouping for scalable secure multicasting. Group security agents share a secret key with each of their subgroup members. Similarly a group security controller distributes a secret key to the top level subgroup. All these keys serve the purpose of key encrypting keys while session keys are distributed during multicast data transfers.

The Centralized Tree-Based Key Management (CTKM) scheme has been proposed separately by three research groups [4, 14, 15]. Members of the multicast group are leaf nodes of the key distribution tree. Each member shares a unique KEK with the group manager. In addition, each member receives all the KEKs corresponding to the internal nodes in its path to the root of the tree. The root's key is the traffic encrypting key (TEK).

Distributed flat key management. This protocol is a distributed version of CFKM-DMG described earlier in this section (refer to [4]). There is no group manager in this scheme. Each member of the multicast is trusted and no one holds/creates more than one traffic encrypting key and W key encrypting keys where W is the number of bits in the binary representation of member IDs. Members joining early generate the keys and are called *key holders*. The ones joining late, receive keys from these key holders.

Tabular comparison. In summary, we list the merits and the shortcomings of existing secure scalable multicasting protocols in Table 5. We first define the terminology used in the table of comparison. In particular, n denotes the number of members in the multicast group, l represents the number of subgroups in the group, and $\bar{l}$ denotes the average subgroup size. c denotes the number of children of the sender of the multicast data and d represents the degree of a hierarchical distribution tree. The inter-relations between these parameters are $0 \leq \bar{l} \leq d$, $l \approx \frac{n}{d-1}$ and $1 \leq c \leq d$.

Note that we list CTKM in the table as a representative for CFKM-DMG and the distributed flat scheme, which are variations of CTKM. The hierarchical scheme uses the largest number of keys while the flat schemes use fewer keys. In all these schemes the session key is modified each time a host joins/leaves. Some of the KEKs are also changed. The number of KEKs updated is different in each of these protocols. Consequently, all these protocols suffer from 1 affects n scalability problem. Also, the distributed flat scheme is vulnerable to security attacks from members of the group. Both the flat schemes also cannot exclude colluding members [4].

4. Conclusions

In this paper, we proposed a dual encryption scheme for scalable secure multicasting. We use hierarchical subgrouping to support scalability. Group control authority is delegated to subgroup managers. We use two sets of key encrypting keys to provide the capability to deny access of multicast data to the subgroup managers. Members of the multicast group receive both the key encrypting keys. Subgroup managers also need to join the group to get access to multicast data. Alternatively, they can participate in the multicast assisting in key distribution and access control.

Table 5. Comparison of secure multicast protocols

	ETM	SMKD	GKMP	PDKD	Iolus	CTKM	**DEP**
Access control mechanism	Yes	Yes	Yes	No	Yes	Yes	Yes
Capability certificate	No	No	Yes	No	No	No	Yes
No. of keys in the multicast group	$n+1$	2	$2n$	$n+1$	$n+l+1$	$\frac{dn-1}{d-1}$	$n+l+1+c$
$\approx$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
No. of keys managed by the sender	2	2	$2n$	2	2	$\frac{dn-1}{d-1}$	$c+2$
No. of keys at a member	1	2	2	2	3	$O(\log_d n)$	4
No. of keys at an SGM	—	2	—	—	4	—	5
Public key/ Secret key	Public	Secret	Secret	Secret	Both	Secret	Both
Join scalability	Yes	Yes	Yes	No	Yes	Yes	Yes
Leave scalability	Yes	No	No	No	Yes	Yes	Yes
Data transmission scalability	No	Yes	Yes	Yes	Yes	Yes	Yes
1 affects n scalability	Yes	No	No	No	Yes	No	Yes
No. of messages at join	$O(1)$	$O(1)$	$O(1)$	$O(n)$	$O(1)$	$O(\log_d n)$	$O(1)$
No. of messages at leave	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(\bar{l})$	$O(d\log_d n)$	$O(\bar{l})$
Total key encryptions during data transmission	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(l)$	$O(1)$	$O(l+c)$
No. of key encryptions at the sender	$O(n)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(c)$
Intermediate nodes	—	Trusted	—	—	Trusted	Not trusted	Not trusted

n: number of members l: number of subgroups $\bar{l}$: average size of a subgroup d: degree c: size of the sender's subgroup

We are working on the development of a prototype of our secure multicasting framework. We plan to use the prototype to quantify the appropriate subgroup size, the number of subgroups and the number of key encrypting keys based on group size and the physical distribution of members in the group. The next phase is to extend our protocol to support many-to-many secure group communication.

References

[1] T. Ballardie. Scalable Multicast Key Distribution. RFC 1949, May 1996.

[2] T. Ballardie and J. Crowcroft. Multicast-specific Security Threats and Counter-measures. In *Proc. Symposium on Network and Distributed System Security*, pages 2–16, San Diego, California, February 1995.

[3] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. Information and Computation, December 1997.

[4] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner. Efficient Security for Large and Dynamic Groups. Technical Report TIK Technical Report No. 41, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, February 1998.

[5] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[6] R. Ganesan. The Yaksha Security System. *Communications of the ACM*, 39(3):55–60, March 1996.

[7] L. Gong. Enclaves: Enablins Secure Collaboration Over the Internet. *IEEE Journal on Selected Areas in Communications*, 15(3):567–575, April 1997.

[8] L. Gong and N. Shacham. Elements of trusted multicasting. In *Proc. IEEE Intl. Conf. on Network Protocols*, pages 23–30, Boston, MA, USA, October 1994.

[9] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) Architecture. RFC 2094, July 1997.

[10] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

[11] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proc. ACM SIGCOMM*, pages 277–288, Cannes, France, September 1997.

[12] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Crypto Systems. *Communications of the ACM*, 21:120–126, 1978.

[13] W. Stallings. *Network and Internetwork Security*. Prentice-Hall Inc., 1995.

[14] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architecture. IETF Draft, July 1997.

[15] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. In *Proc. ACM SIGCOMM*, August 1998.